Thomas Bayes aurait-il dû croquer le fruit défendu?

Guillaume Poirier-Morency
Département d'informatique et de recherche opérationelle
Université de Montréal
Montréal
guillaume.poirier-morency@umontreal.ca

Gabriel Lemyre
Département de mathématiques et statistiques
Université de Montréal
Montréal
gabriell@dms.umontreal.ca

Abstract

Il existe différents types de données qui exhibent des relations particulières entre les dimensions. Nous comparons ici l'efficacité de différents modèles d'apprentissage sur deux ensembles de données: les caractères manuscrits de l'échantillon MNIST et les données de prédiction de salaires. En particulier, nous nous intéressons au classifieur de Bayes, aux arbres de décisions et au perceptron multi-couche. Nous explorerons différents pré-traitement pour mesurer les gains possibles lorsque combinés avec un modèle traditionnel de classification. Notre intuition nous porte à croire que les méthodes de types arbres de décision et classifieur de Bayes seront plus efficaces sur les prédictions de salaire que sur MNIST. Nous expérimentons aussi avec un modèle de Bayes mixte. Cette intuition est justifiée par le fait que les attributs pour les salaires sont plus clairement scindés. Pour contourner le fait que certains des attributs de l'échantillon de prédiction de salaire sont de type catégorielle, nous prévoyons effectué une transformation de type *one-hot* et ainsi considérer des vecteurs numériques plutôt que des classes.

Analyses préliminaires

Les attributs de l'échantillons de prévision de salaires étaient au nombre de treize (13) et étaient séparables en attributs catégoriels et attributs continus :

Continus
- Age
- Financial weighted
- Capital gains
- Capital loss
- Hours per week

Catégoriels
- Work class
- Education
- Education code
- Marital status
- Occupation
- Relationship
- Race
- Sex
- Native country

Nous avons choisis de considérer que l'entrée *Education* et de laisser tomber *Education code*, puisque les deux font référence à la même chose. Nous traiterons donc douze (12) attributs.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Pour traiter ces attributs, il est d'abord important de prendre une décision en ce qui a trait à l'imputation des données manquantes. Puisque les algorithmes de type arbres de décisions et classifieur de Bayes ne prennent pas de données manquantes, nous avons choisis de faire les remplacements suivants :

Attributs *continus* - **Moyenne** empirique sur D des valeurs de cet attribut
Attributs *catégoriels* - **Mode** empirique sur D des valeurs de cet attribut

Pour traiter les données catégorielles, il était important de les transformer, tel que discuté dans le résumé, en vecteur *"one-hot"*. Nous obtenons donc un total de 99 attributs binaires, cinq (5) attributs continues et une (1) valeur binaire pour la sortie (ou cible).

Les figures en annexes montrent que la tâches de classification des données de salaires n'est pas triviale. En effet, pour les attributs continus (figure 10), une analyse par paires d'attributs ne permet pas d'entrevoir la possibilité d'une séparabilité linéaire des données. La même hypothèse est faites en observant les histogrammes correspondants aux attributs catégoriels (figure 9) puisqu'aucun attribut ne permet de séparer parfaitement les entrées. Quelques valeurs des attributs catégoriels semblent permettre de trancher, ceci laisse croire que l'utilisation d'arbre de décision est justifiée.

Méthodologie

Les données de salaire ont été imputés avec le mode pour les caractéristiques catégoriques et la moyenne pour celles continues.

Nous avons utilisé les bibliothèques scikit-learn (Pedregosa et al. 2011) et Keras (Chollet et others 2015).

Toutes les opérations ont été effectuées à l'aide d'un `Pipeline` [1] qui permet d'assembler les opérations (i.e. imputation, codage *"one-hot"*) dans une chaîne de montage.

La recherche par grille fournie par la classe `GridSearchCV` [2] a été utilisée pour déterminer les meilleurs hyper-paramètres. Nous avons également utilisé un super calculateur pour paralléliser ce procédé.

Les réseaux de neurones ont été hyper-paramétré manuellement et seulement l'époque optimale a été déterminée par validation croisée.

Nous avons utilisé l'algorithme Adadelta (Zeiler 2012) pour faire l'optimisation qui utilise une moyenne exponentielle des gradients des étapes précédentes.

Nous avons également favorisé l'approche Dropout (Srivastava et al. 2014) au lieu des régularisations L1/L2 puisqu'elle semblait bien fonctionner empiriquement. Cette méthode consiste à rendre inneffective une proportion aléatoire d'unités dans une couche de neurones.

Les courbes d'apprentissages affichent l'erreur de classification en fonction de la valeur d'un hyper-paramètre en considérant les valeurs des autres hyper-paramètres qui minimisent l'erreur de validation. Cela fait en sorte que le minimum observé correspond au vrai minimum.

Classifieurs de Bayes

Nous avons expérimenté trois variantes du classifieurs de Bayes:

— noyau Gaussien;
— noyau de Bernoulli;
— variante mixte.

La variante mixte combine les log-probabilité à postériori de chacun des modèles de la manière suivante:

$$\log \Pr[c|X_{cat}, X_{cont}] = \lambda(\log \Pr[c|X_{cat}]) + (1 - \lambda)(\log \Pr[c|X_{cont}])$$

1. http://scikit-learn.org/0.18/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline
2. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Nous remarquons que la variante mixte performe particulièrement bien sur les données de salaires. Ce qui est remarquable est qu'elle est significativement meilleure que les deux modèles pures (i.e. chaque extrémités du graphe).

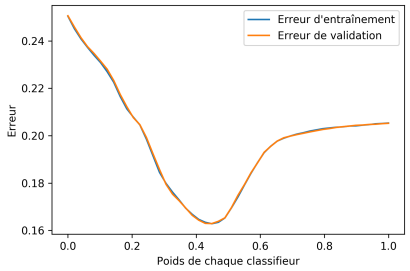


FIGURE 2 – Courbe d'apprentissage du classifieur Bayésien mixte pour le paramètre λ sur les données de salaire

Une hypothèse pouvant expliquer cette performance supérieure est que chaque méthode comporte des avantages sur certains types d'attributs. Le noyau gaussien est fort probablement avantageux sur les attributs continus et le noyau de Bernoulli n'est actif que lorsque les attributs sont catégoriels. Ainsi plus d'information implique plus de performance.

La mauvaise performance du noyau Bernoulli est potentiellement explicable par le fait que les entrées du vecteur *"one-hot"* sont mutuellement exclusives et donc corrélées ce qui pose problème avec notre utilisation du classifieur naïf.

L'autre aspect intéressant est que sa courbe d'apprentissage est identique pour l'entraînement et la validation, ce qui est cohérent avec le fait que les modèles Bayésiens ont une très faible capacité.

Puisque le classifieur de Bayes naïf fait l'hypothèse d'indépendance, si dans un sous-ensemble au moins une valeur possible d'un des attributs n'est pas observée, le calcul du produit des densités sur chaque dimensions de l'entrée retournera zéro (0) et l'entrée en question perdra tout son poids dans le calcul. Pour contourner ce problème, nous utilisons le lissage Laplacien. Cette méthode ajoute simplement une constante Δ à tous les comptes lors du calcul de la fréquence de chaque classe. Il est très important que cet ajout ne soit pas significatif par rapport à la valeur de compte la plus faible observée avant le lissage.

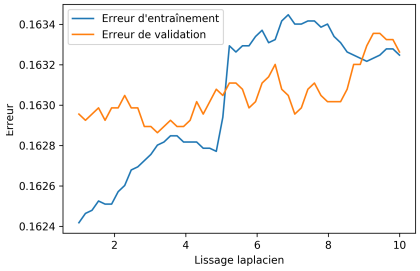


FIGURE 3 – Courbe d'apprentissage du classifieur Bayésien mixte pour le lissage Laplacien sur les données de salaire

Pour les données de MNIST, nous avons testé le classifieur de Bayes à noyau Gaussien ainsi que celui de Bernoulli en arrondissant les degrés de gris à des valeurs binaires. Le taux de bonnes classifications est particulièrement fort: 83.18%!

Nous sommes portés à croire que puisque les images sont centrée et normalisée, les similarités entre les exemplaires d'une même classe se traduisent par un ensemble de pixels communs assez stable. Cette signature est facilement reconnue par une distribution conjointe de succès. Cette particularité sera discutée plus en détails dans la dernière section.

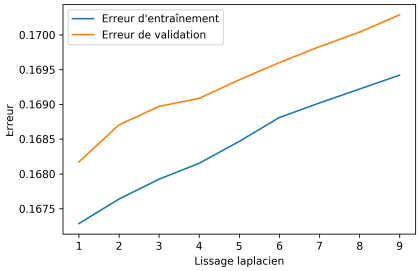
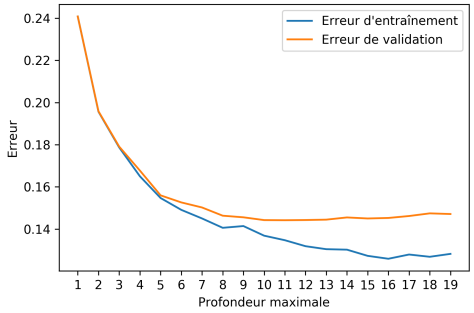


FIGURE 4 – Courbe d'apprentissage du classifieur Bayésien à noyau Bernoulli sur MNIST

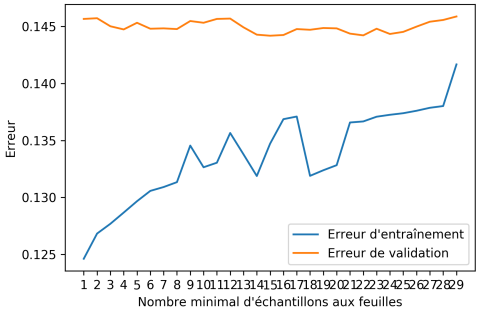
Arbres de décisions

L'utilisation des arbres de décision est habituellement appropriée dans le cas de données dont les attributs présentent des caractéristiques de haut niveau. Les données de prédiction de salaires présentent de tels attributs, mais les points de MNIST (en niveaux de gris) n'ont pas une représentation de haut niveau ce qui explique peut-être la faible performance de cet algorithme pour la classification des images.

Modifier la profondeur maximale permet de contrôler la capacité et la propension de l'algorithme au sur-apprentissage. Ceci est observable sur les figures 5a et 6a. On remarque en fait qu'à partir d'une

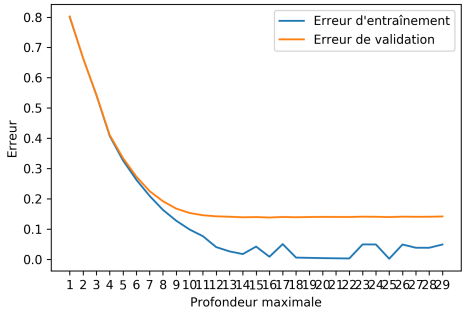


(a) HP: Profondeur de l'arbre

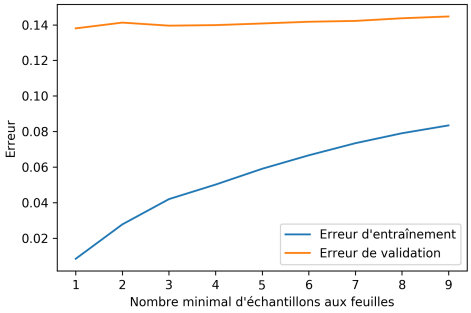


(b) HP: Nombre d'observation minimale par feuille

FIGURE 5 – Courbe d'apprentissage pour Arbres de décision sur la prédiction de salaires



(a) MNIST HP: Profondeur de l'arbre



(b) MNIST HP: Nombre d'observation minimale par feuille

FIGURE 6 – Courbe d'apprentissage pour Arbres de décision sur MNIST

certaine profondeur, l'erreur d'entraînement diminue alors que celle de validation stagne, indiquant un sur-apprentissage.

Une augmentation du nombre minimal d'observation par feuille implique une diminution de la capacité du modèle. À l'extrême, sans limiter la profondeur maximale de l'arbre, permettre qu'il n'y ait qu'un exemplaire par feuille donnerait un arbre où il existe un chemin menant à chaque exemplaire de l'ensemble d'entrainement. Ce cas constiturait un exemple type de sur-apprentissage que l'on peut observer aux figures 5b et 6b.

Les arbres de décisions sont des modèles à très forte capacité et la profondeur maximale est définitivement l'hyper-paramètre contrôlant le mieux la capacité du modèle.

Perceptron multi-couche

Le perceptron multi-couche ne convergait pas sur les données de salaire. Nous avons tout de même essayé plusieurs architectures, régularisations et même l'ajout de couches cachées. Les résultats de la figure 7 illustre bien que le réseau ne s'améliore pas vraiment au fur et à mesure des époques.

La descente en escalier de la figure 8a semble être liée à la régularisation Dropout: à chaque fois qu'un neurone est neutralisé (i.e. la perte stagne), la rétro-propagation tente de trouver une nouvelle façon de faire baisser la perte.

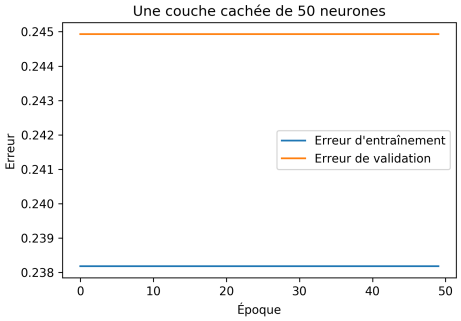
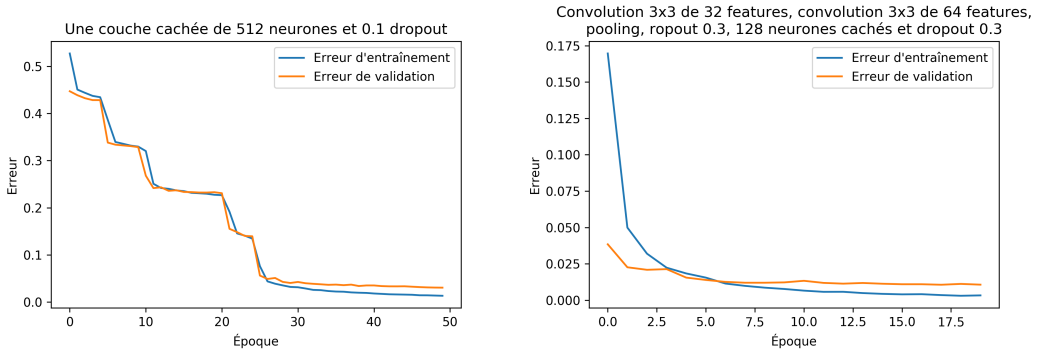


FIGURE 7 – Courbe d'apprentissage du perceptron multi-couches sur les données de salaire

Tel qu'anticipé, le réseau de neurones convolutif est très performant et convergent rapidement. L'approche est très intéressante: k noyaux de convolution sont appliqués sur chaque région de l'image pour former un ensemble de représentation intermédiaire en ensuite une mise en commun (*pooling*) est effectué pour écraser ces attributs dans une représentation compacte. Ensuite, la représentation en 2 dimensions est transformée en un vecteur sur lequel on applique un réseau de neurones traditionnel.



(a) **perceptron multi-couches** (b) Réseau de neurone **convolutif**

FIGURE 8 – Courbe d'apprentissage des réseaux de neurones sur MNIST

Un des avantages de ce type de modèles est qu'il réutilise les même poids pour chaque noyau de convolutions, ce qui réduit considérablement le temps d'entraînement permet d'exploiter les caractéristiques de localité de l'image.

Résultats finaux et discussion

Les tableaux suivants correspondent aux valeurs de précision sur les ensembles de validation et de test pour les données de prédiction de salaires.

TABLE 1 – (*Résultats finaux Prédiction de salaire*)

Salary	Validation	Test
Bayes naïf mixte	83.81%	73.01%
Arbres de décisions	85.57%	75.82%
Perceptron multi-couche	75.45%	76.37%

En générale, on remarque que les modèles utilisés n'ont pas très bien performé à l'étape de test. Ceci est très surprenant pour le modèle de Bayes qui ne devrait pas avoir souffert de sur-apprentissage.

Nous observons que l'échantillon complet recuilli est déséquilibré, présentant environ 75% de cibles $<= 50K$ et 25% de cibles $> 50K$. Nous avons construit les sous-ensembles d'entrainement, de validation et de test selon ces mêmes proportions. En analysant les classifications finales sur l'ensemble de validation pour les arbres de décision, nous obtenons les valeurs du tableau 2.

TABLE 2 – (*Rapport de classification DT*, **Validation**)

Cible	precision	recall	f1-score	support
$<= 50K$	0.88	**0.95**	0.91	24720
$> 50K$	0.79	**0.59**	0.67	7841
avg/total	0.86	0.86	0.86	32561

La colonne "*recall*" fait référence au nombre de points de cette classe qui furent bien classés. Il est évident que l'algorithme est nettement meilleur pour identifier les exemplaire ayant pour cible $<= 50K$. En analysant le rapport de classification sur l'ensemble de test, nous observons que ce déséquilibre est encore plus fort.

TABLE 3 – (*Rapport de classification DT*, **Test**)

Cible	precision	recall	f1-score	support
$<= 50K$	0.81	**0.89**	0.85	12435
$> 50K$	0.48	**0.33**	0.39	3846
avg/total	0.73	0.76	0.74	16281

Les critères utilisés pour la selection des tests aux noeuds de l'arbre supposent souvent une distribution uniforme des classes des exemplaires y étant traités. Ainsi, un déséquilibre entre la fréquence de chaque classe à un noeud pourrait entrainer une préférence des tests pour une classe en particulier. C'est peut-être ce que nous observons avec les données de prédiction de salaires. L'utilisation de l'enthropie décentrée (« Arbre de décision pour données déséquilibrées » 2017) comme mesure de désordre serait une solution potentielle à étudier dans une analyse subséquente de cet échantillon puisque cette méthode pondère en fonction de l'importance de chaque classe aux noeuds.

Il serait aussi intéressant de discrétiser les attributs continus pour tester nos classifieur de Bayes et arbres de décision sur des attributs discrets seulement.

Il était attendu que les réseaux de neurones ne soient pas optimaux sur les prédictions de salaires à cause de la quantité élevée d'attributs catégoriels mélangés aux attributs continus. La performance sur l'ensemble de validation et de test est cependant très comparable ce qui confirme que les réseaux de neurones ne sont pas très sensible aux problèmes de sur-apprentissage. En analysant la figure 7, il ne semble pas que d'augmenter le nombre d'époque est le moindre effet sur la précision du réseau. Les résultats de la table 1 mettent en évidence le même type de problème conformément à la répartition des classes dans l'échantillon initial en ce qui a trait à la précision des réseaux de neurones.

Pour ce qui est des résultats sur MNIST, la performance obtenue est relativement conforme à notre intuition.

TABLE 4 – (*Résultats finaux MNIST*)

MNIST	Validation	Test
Bayes naïf gaussien	56.28%	55.77%
Bayes naïf de Bernoulli	83.18%	83.36%
Arbres de décisions	86.19%	87.37%
Perceptron multi-couche	96.93%	96.86%
Réseau de neurones convolutif	98.77%	98.87%

En général, les modèles ont bien performé sur l'ensemble MNIST. Nous sommes étonnés de voir l'erreur de test diminuer pour les arbres de décision, quoique le taux de classification n'est peut-être pas suffisament élevé pour que ce soit significatif.

Le positionnement des images au centre et leurs dimensions sensiblement stables a beaucoup contribué à la performance des classifieur de Bayes naïf. Ceux-ci ayant pour objectif de trouver une "distribution" pour analyser les pixels auraient probablement très mal performé sur des images de caractères non transformées. Les applications de ce type d'algorithme dans la monde "réel" serait donc très limitées sans l'ajout d'une transformation a priori sur les images pour normaliser la position et l'échelle des caractères.

Les modèles de réseaux de neurones ont très bien conservé leur performance assez élevée sur cet ensemble de données.

La position et l'échelle n'est pas aussi significatif pour les réseaux convolutifs qui analyse de petites fenêtres de l'image. Ceux-ci pourraient fort probablement déceler les subtilités des caractères mêmes

si leur taille ou positionnement n'est pas constant. Leur faible sensibilité aux variations de ces deux facteurs explique peut-être en partie leur performance plus élevée que les réseaux de neuronnes de type MLP.

Nous avions prévu tester la performance d'arbres de décision dont les noeuds prennent une décision en fonction d'une combinaison linéaire des attributs obtenue par une PCA. Le temps nous ayant malheureusement manqué, il serait très intéressant de tester ce type d'algorithme dans un projet subséquent. Nous espérons ainsi augmenter la performance de ces arbres sur des échantillons semblable à la prédiction de salaire où ces algorithmes semblent souffir de sur-apprentissage.

Répartition

Guillaume s'est occupé de la programmation et Gabriel de la rédaction de la présentation du projet et du rapport. Nous avons fait l'analyse exploratoire des données ensemble.

Annexe

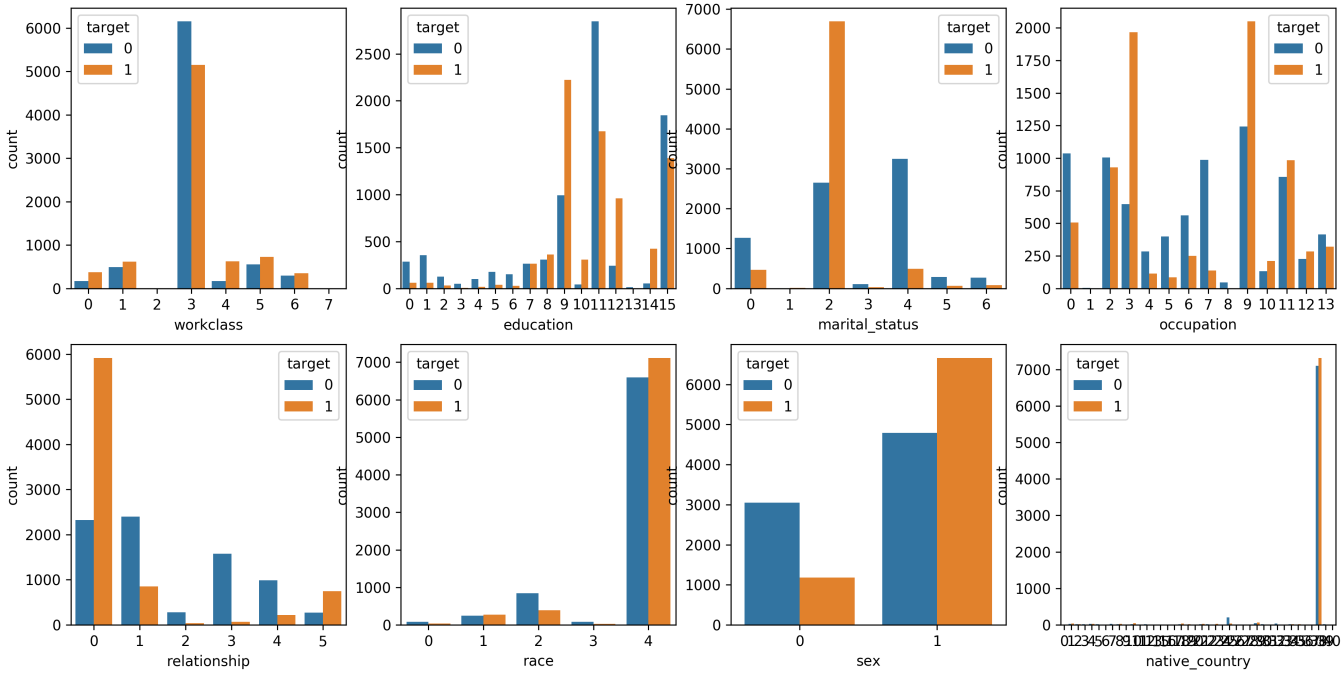


FIGURE 9 – Histogramme des données catégorielles en fonction de la cible associée

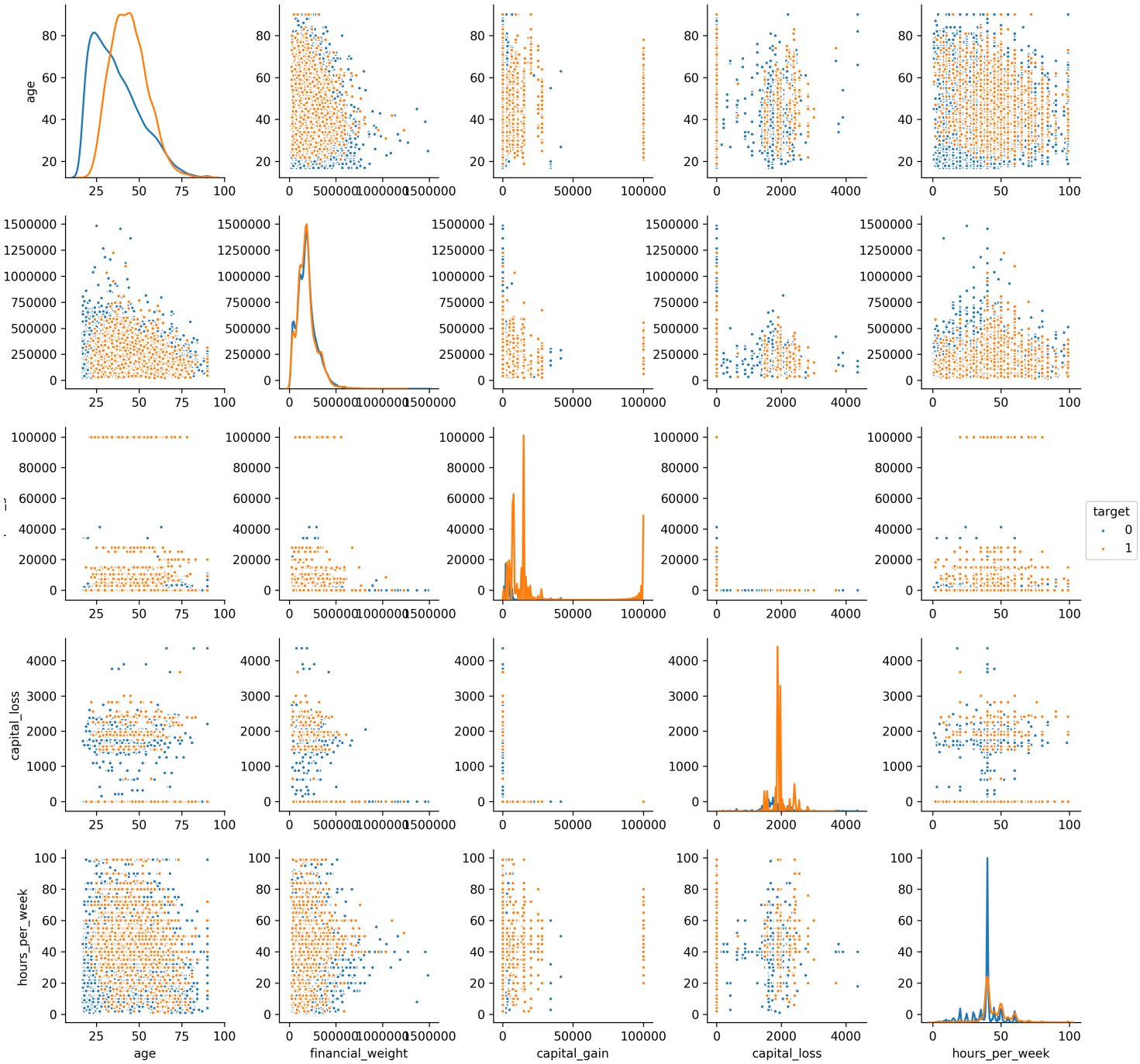


FIGURE 10 – Analyse par paires d'attributs continus

Références

« Arbre de décision pour données déséquilibrées ». 2017. Consulté le décembre 22. http://mephisto.unige.ch/pub/publications/gr/RitsMarZigh_ArbreImplic.pdf.

Chollet, François, et others. 2015. « Keras ». https://github.com/fchollet/keras; GitHub.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. « Scikit-learn: Machine Learning in Python ». *Journal of Machine Learning Research* 12: 2825-30.

Zeiler, Matthew D. 2012. « ADADELTA: An Adaptive Learning Rate Method ». *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.